УДК 004.056.55:658.262

IMPLEMENTATION AND COMPLEXITY ANALYSIS OF EMBEDDED ADVANCED ENCRYPTION STANDARD (AES) ENCRYPTION IN BLUETOOTH LOW ENERGY COMMUNICATION

Sarp Erturk, ***Tuvshinjargal Ulziiutga***, *Master, Kocaeli University, Department of Electronics and Communications Engineering, Kocaeli city, Turkey. Phone: 0090 553 530 95 57, e-mail: hw09d004@gmail.com*

Abstract

This article presents and implementation and complexity analysis of embedded advanced encryption standard (AES) encryption in Bluetooth low energy, applied on the communication between an embedded system, in this case the STM32F4 Discovery board, and a smart phone. The paper covers time complexity of the advanced encryption standard applied to transmitted Data via Bluetooth low energy and analyzes how much time has been spent in the encryption and transmission phases respectively. The Bluetooth low energy protocol, has been made available to

send Data over short distances with low energy and is a rapidly developing technology for new products on the market. For example, Bluetooth low energy loudspeakers and heart beat measurement devices for human health are typicall applications that require the transmission between an embedded system and a smart device (phone/tablet). In partscular for applications like health, the security and integrity of Data is of upmost importance and this is where encryption can be very useful.

Keywords: Bluetooth Low-Energy, BLE, Data Encryption, AES (Advanced Encryption Standard), STM32F4, Android

ВВОД В ДЕЙСТВИЕ И ТРУДНОСТИ ПРИ ПРОВЕДЕНИИ АНАЛИЗА ВСТРОЕННЫХ РАСШИРЕННЫХ СТАНДАРТОВ ШИФРОВАНИЯ (AES) В BLUETOOTH С НИЗКИМ ЭНЕРГОПОТРЕБЛЕНИЕМ

Сарп Эртюрк*, *Туфшинжаргал Улзиитга, *Магистрант, Университет Коджаэли, Отдел электроники и инженерных коммуникаций, Коджаэли, Турция. Тел: 0090 553 530 95 57, e-mail: hw09d004@gmail.com*

Аннотация

В данной статье описываются трудности при проведении анализа встроенных расширенных стандартов шифрования (AES), применяемые для передачи данных между платой разработки STM32F4 и смартфоном с помощью Bluetooth с низким энергопотреблением. Кроме того, описывается временная сложность расширенного стандарта шифрования, используемая при передаче данных с помощью Bluetooth с низким энергопотреблением, а также, количество времени, требуемое для этого. Протокол Bluetooth с низким энергопотреблением предназначен для передачи данных на короткие расстояния и используется во всем мире, а также является быстро развивающейся технологией для нового выхода на рынок. Например, динамик Bluetooth с низким энергопотреблением, устройство для измерения пульса, необходимое для здоровья человека применяются для передачи данных между встроенной системой и смарт девайсом любого вида (телефон/таблетка). В частности, для приложений, таких как здоровье, безопасность и целостность данных, шифрование имеет первостепенное значение.

Ключевые слова: Bluetooth, BLE, Шифрование данных, AES, STM32F4 Discovery, Android Bluetooth API

1. Introduction

An important part of advancement in technology has been in the field of wireless Data communication. This paper focuses on an important application of wireless Data transmission; in that the Advanced Encryption Standard (AES), that is one of the state-of-the-art Data encryption methods, is applied to Bluetooth Low Energy communication for secure Data transfer.

Bluetooth was first presented by Ericsson Telecommunications in 1994 to transmit Data using short-range radio frequency transmission in the 2.4GHz band[1]. Since then Bluetooth has found its way into many consumer products inclusing smart phones and tablets. Quite recently, in 2010 Bluetooth low energy (BLE) was introduced, aiming to consume less energy [2]. Any kind of BLE equipment is aimed to connect and work with any supported device including smart devices. As a result devices that are constantly connected with smart phones have been developed because they require low power operating at short distances. Examples are the control of a robot arm[3] and a Bluetooth based heartbeat monitor[4].

However, Data transferred using the conventional BLE standart is not really secure and can be easily accessed as it is transferred over air. There encryption is applied to the transferred Data on

BLE in this paper to provide end-to-end security. This paper presents the implementation of an AES based encrypted BLE transmission between an embedded system and a smart device. Furthermore an analysis is carried out to determine the time spend for processing encrypted Data versus unencrypted Data.

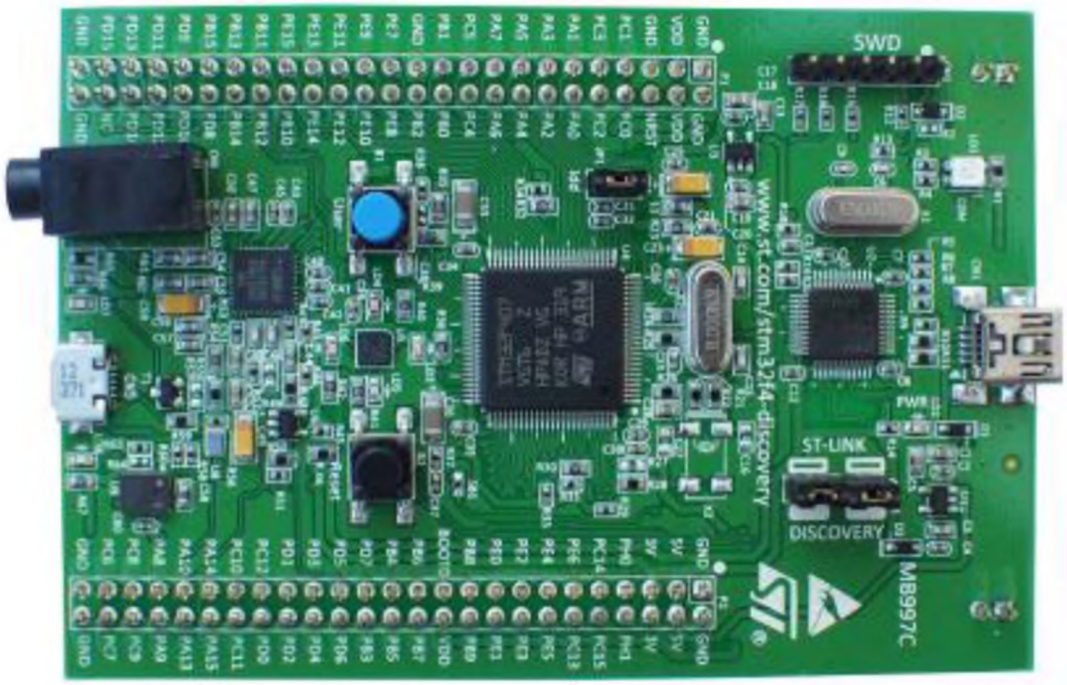
The second section of this paper contains information about the general architecture of the system. In the third section, the structure of the AES standard as applied in the project is covered. The fourth section provides information about the main algorithm and sotfware developed when applying AES in the project. In section five, timing analysis is provided to determinde the introduced complexity by encoding. The sixth and last section provides conclusions.

2. System architecture

This paper presents the implementation and analysis of BLE communication with AES encryption between an embedded system and a smart device. AES encryption increases security, in particular as the communication is wireless and can be intercepted over air.

In the implementation presented in this paper the embedded system is implemented on the STM32F4 Discovery board. The STM32F4 Discovery board shown in Figure 1 has a STM32F407VGT ARM 32-bit based microcontroller [7]. This microcontroller has 100 digital general purpose input / output (GPIO) ports. The Discovery board enables easy utilization of the processor for embedded system development with features like 17 timers, 3 ADCs, 1 MB FLASH with 192 + 4KB RAM, microcontroller running at 168 MHz with 6 USART interfaces.

Figure.1: STM32F4 DISCOVERY[8].



For BLE capability the TinySine BLE module as shown in Figure 2 is used[9]. This module is capable of operating in dual mode to transmit and receive Data between the STM32F4 micro controller and a smartphone. This module operates at 3.3V and is therefore compatible for utilization with the STM32F4 Discovery board. The USART interface is used to connect the Bluetooth module to the STM32F4 processor.

FIGURE.2: TinySine dual Bluetooth module[9].

An Android based smart phone is used on the other end for Bluetooth communication. Hence, the communication takes places between a STM32F4 based embedded system and a smart device. This setup has many implementation possibilities for real world applications.

3. Advanced encryptıon standard

The advanced encryption standard (AES) is a symmetric Data encryption algorithm[5]. This algorithm is developed by Joan Daemen and Vincent Rijmen[6]. In the AES algorithm, the input and output Data is typically a 128-bit Data matrix[5]. The AES algorithm can present 128-bit Data blocks with 128, 192 or 256 bit key options[5]. The AES implementation specifies the length of the key, the number of laps of the cipher and how many times to loop. A 128 bit key was used in the implementation presented in this paper.

The input Data is placed in a 4×4 byte Data matrix that is called the state matrix. Besides the state matrix, there is a 16-byte (i.e. 128-bit) key used for encrypting the Data, and the operations are performed on both the state matrix and the 16-byte key.

The AES implementation can be summarized as[2]:

• Key expansion: Construct 11 round keys (sub-keys) of 128 bit length from the 128 bit main AES key.

• Initial Round:

1. Add Round Key: XOR each byte of the state matrix with a block of the round key

• Rounds:

1. Sub-bytes: Each byte of the block in the state matrix is replaced by a byte from a reference S-table. So; instead of the original hexadecimal value of the Data, the new (replacement) hexadecimal value is set
2. Shift Rows: The last three rows of the state matrix are shifted cyclically a certain number of steps. The first row is not shifted, the second row is shifted by one position, the third row by two and the forth row by three positions.
3. Mix Columns: The four bytes of each column of the state matrix are combined using an invertible linear transform. Together with row shifting, this process provides diffusion in the cipher.
4. Add Round Key

• Final Round:

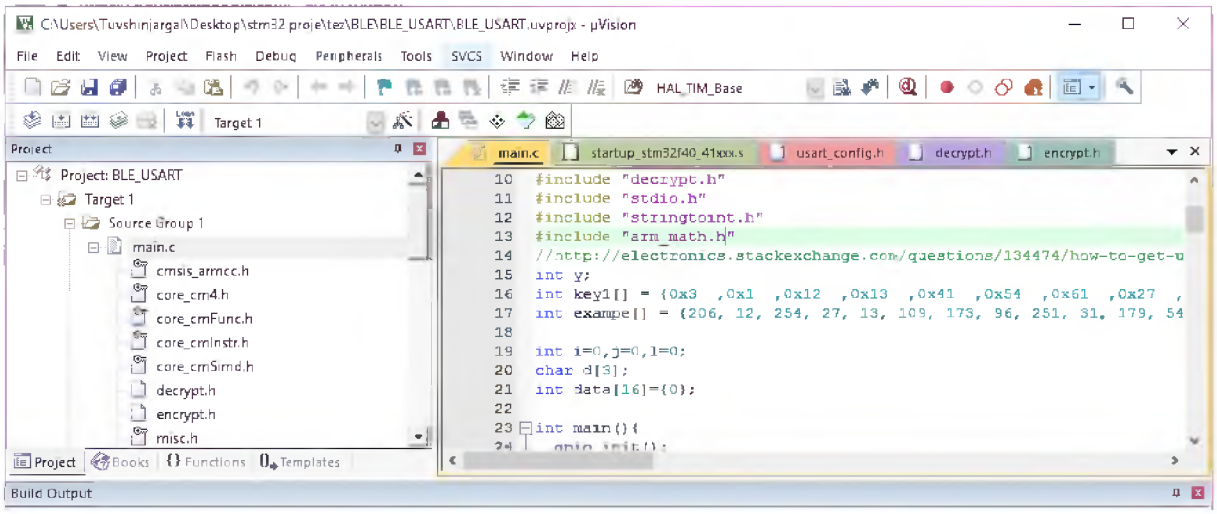
1. Sub-bytes
2. Shift Rows
3. Add Round Key

The rounds are repeated by a specific number of times depending on the key length used in encryption. The number of rounds is 10 when the key length is 128 bits. The pseudo code implementation for this operations is provided in section 4.

4. Software ımplementatıon

The KEIL uVision Integrated Develoment Environment (IDE) is used for the development of the embedded Software. This IDE provides the environment for programming ARM-based microcontroller development boards. The Keil uVision IDE main screen is shown in Figure 3.

Figure 3: Keil uVision main screen.



In the Keil uVison IDE, the settings of the Data input and output of the STM32F4 are selected to use the USART interface so as to interface the TinySine Bluetooth module. Separate Encrypt and Decrypt functions are implemented to facilitate AES encryption and decryption respectively.

The STM32F4 Discovery development board is connected to the Pcthrough its own programmable USB port, so that the code can be downloaded directly into the microprocessor. The flowchart of the embedded Software implementation for the STM32F4 microcontroller is shown in Figure 4a.

Figure 4a: Block diagram of the Software in STM32F4

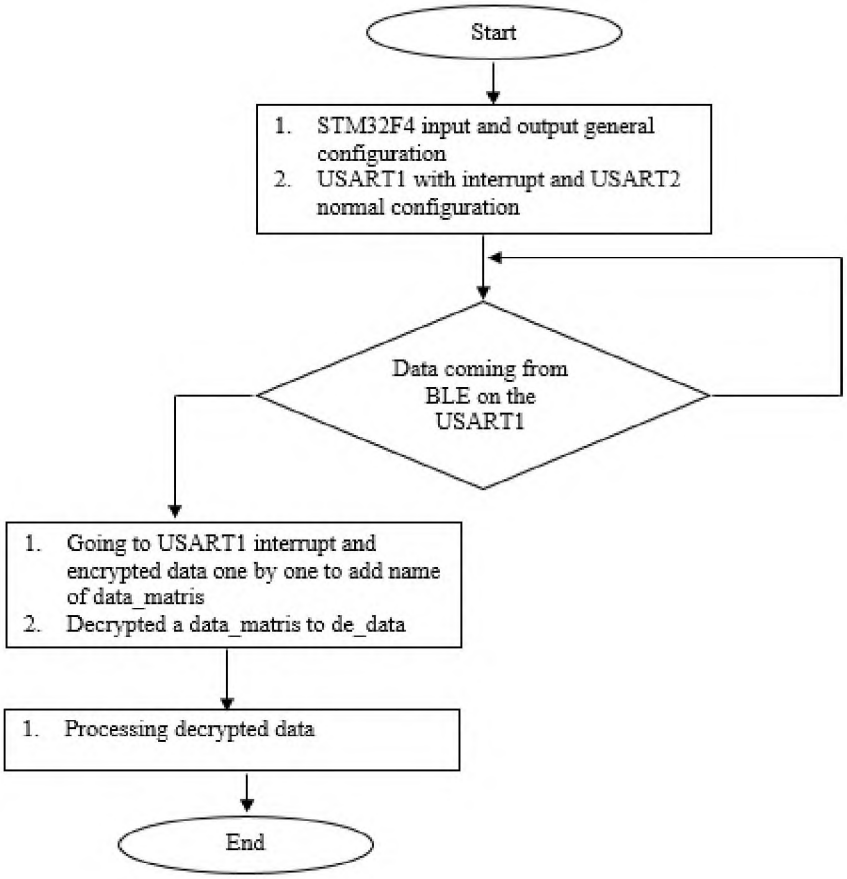
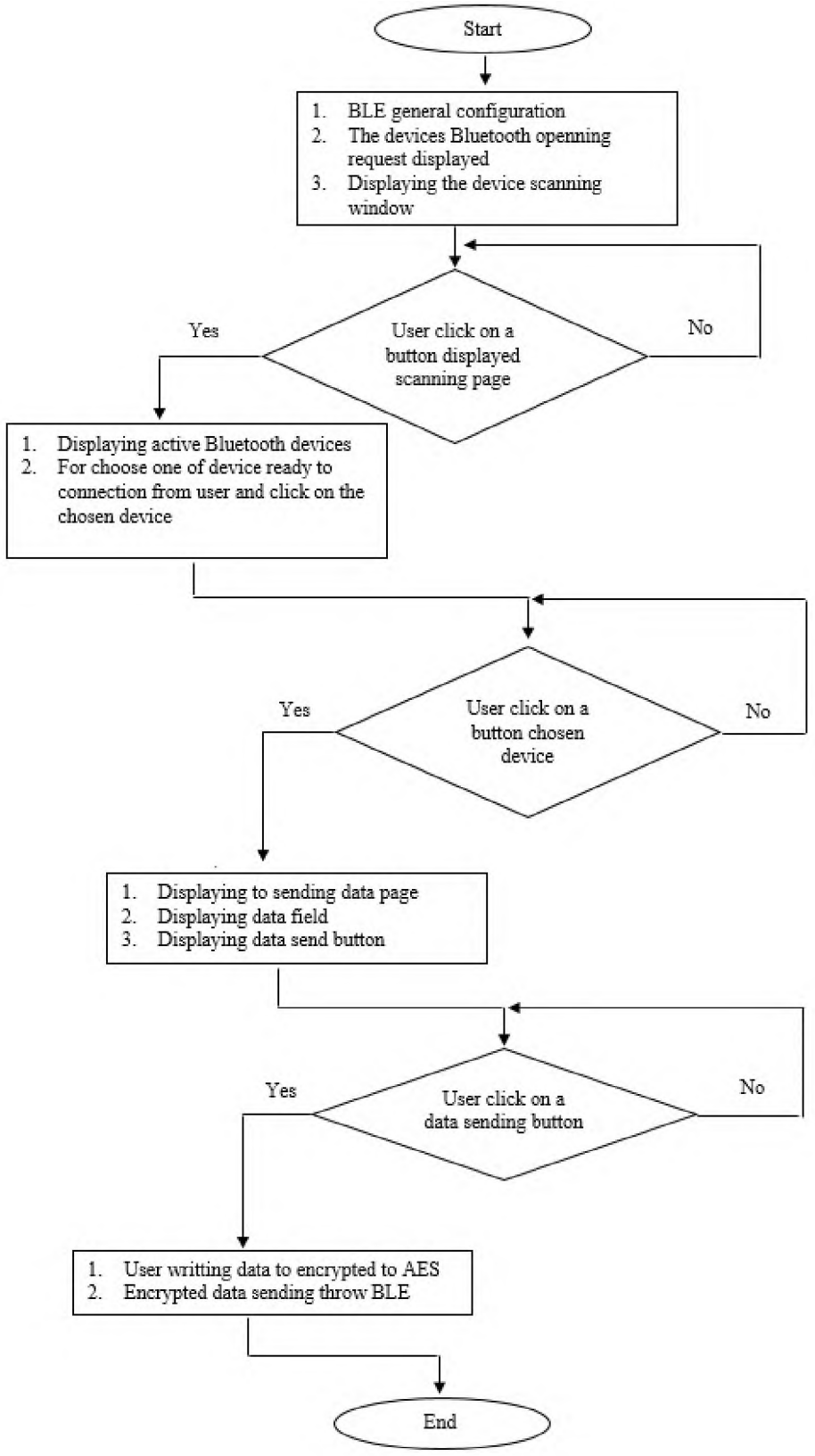


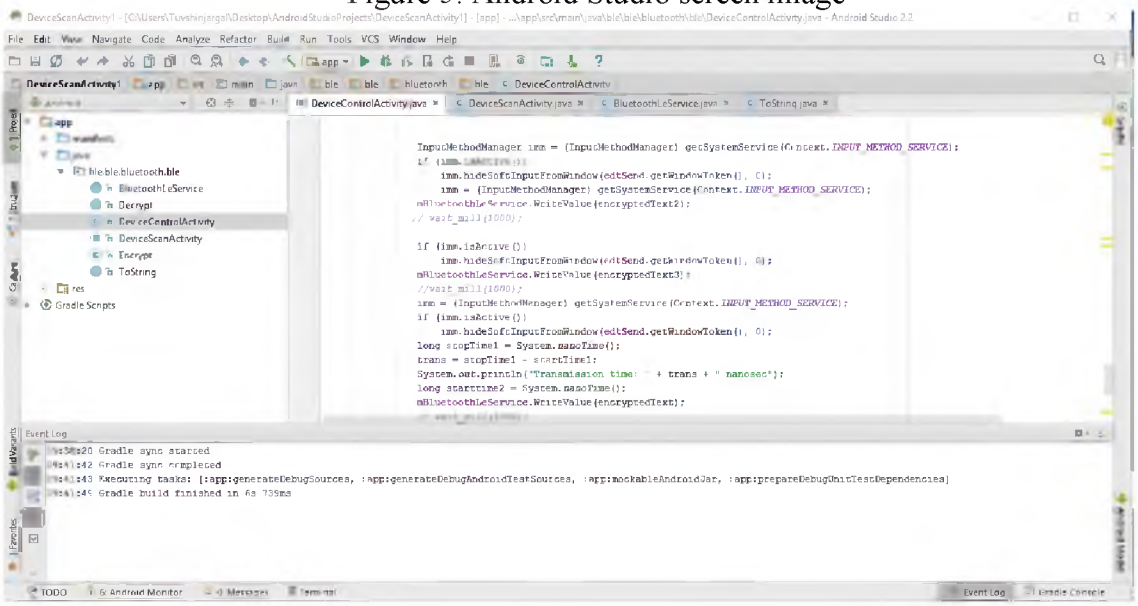
Figure 4b: Block diagram of the Software in Android Studio



The smart device Software implementation is carried out using Android Studio. This development environment facilitates the development of Software to run as an application on Android smart devices. Android Studio, provides a simple user interface for users. The application is basically developed using Java language.

The BLE communication and AES functionalities are implemented in Java language in Android Studio development environment. The functionality is implemented in the form of an Application directly running on the smart device in Android. Hence it can be used in any smart device that runs on Android Software. The Android Studio screen image is shown in Figure 5.

Figure 5: Android Studio screen image



In the application, initial settings for BLE functionality as well as Data encryption and decryption functions are implemented. In addition, Android Studio provides an interface that allows users to see when a bluetooth device is connected for debugging purposes.

The application implements a scan button so that existing Bluetooth devices are scanned and shown to the user to enable the request for connecting to a device. After the connection is made, the "send Data" window is displayed. The user can type any cmessage and press the "send button", so that the message Data is encrypted with the advanced encryption standard and transmitted to the STM32F4 based embedded system. The advanced encryption standard pseudo code is given in Figure 6.

Figure 6: Advanced encryption standard (AES) pseudo code

```
1   Cipher(byte in[]){
2   begin
3       int i,j round=0;
4       for(i = 1 step to 1->4)
5           for(j = 1 step to 1->4)
6               state[j][i] =  in[i*4 + j];
7
8
9       AddRoundKey(0);
10      for(round=1 step to Nr-1)
11          {
12              SubBytes();
13              ShiftRows();
14              MixColumns();
15              AddRoundKey(round);
16          }
17      SubBytes();
18      ShiftRows();
19      AddRoundKey(Nr);
20      for(i = 1 step to 1->4)
21          for(j = 1 step to 1->4)
22              out[i*4+j]= state[j][i];
23
24  end
```

The subfunctions shown in Figure 6 provide the functionality explained in section three, where Nr = 10 because the advanced encryption standard key length is 128 bits. Note that here Nr represents the total number of rounds. The sub-functions at the end of all rounds prepare the Data for the last step.

The key extension function is executed before the Data encryption functions. The pseudo code for the key extension function is shown in Figure 7. The key extension function creates round keys by changing the main AES key step by step. This provides security for the key. In order to decrypt Data, the operations performed in the encryption step are performed in reversed order. In all cases the key extension process is performed before Data encryption and decryption to create the necesary round keys.

Figure 7: Key extension pseudo code

```
keyexpansion(int key[])
begin
    int temp[4];
    int i,j,k;
    for(i=0;i<nk;i++)
    {
        roundkey[i*4]=key[i*4];
        roundkey[i*4+1]=key[i*4+1];
        roundkey[i*4+2]=key[i*4+2];
        roundkey[i*4+3]=key[i*4+3];
    }
    while (i < (nb * (nr+1)))
    {
                    for(j=0;j<4;j++)
                        temp[j]= roundkey[(i-1) * 4 + j];
                    if (i % nk == 0)
                    {
                        {
                            k = temp[0];
                            temp[0] = temp[1];
                            temp[1] = temp[2];
                            temp[2] = temp[3];
                            temp[3] = k;
                        }
                        {
                            temp[0] = getsboxvalue(temp[0]);
                            temp[1] = getsboxvalue(temp[1]);
                            temp[2] = getsboxvalue(temp[2]);
                            temp[3] = getsboxvalue(temp[3]);
                        }
                        temp[0] = (temp[0] ^ rcon[i/nk]);
                    }
                    else if (nk > 6 && i % nk == 4)
                    {
                        {
                            temp[0]=getsboxvalue(temp[0]);
                            temp[1]=getsboxvalue(temp[1]);
                            temp[2]=getsboxvalue(temp[2]);
                            temp[3]=getsboxvalue(temp[3]);
                        }
                    }
                    roundkey[i*4+0] =  (roundkey[(i-nk)*4+0] ^ temp[0]);
                    roundkey[i*4+1] =  (roundkey[(i-nk)*4+1] ^ temp[1]);
                    roundkey[i*4+2] =  (roundkey[(i-nk)*4+2] ^ temp[2]);
                    roundkey[i*4+3] =  (roundkey[(i-nk)*4+3] ^ temp[3]);
                    i++;
    }
    end
```

5. Experimental results and time complexity analysis

This section presents experiments for time complexity analysis for implementing the AES the advanced encryption standard with BLE communications. The experiments were carried out using a Android smart phone with the application being installed using the Android Studio 2.2 debugger.

The total number of operations required for AES encryption is

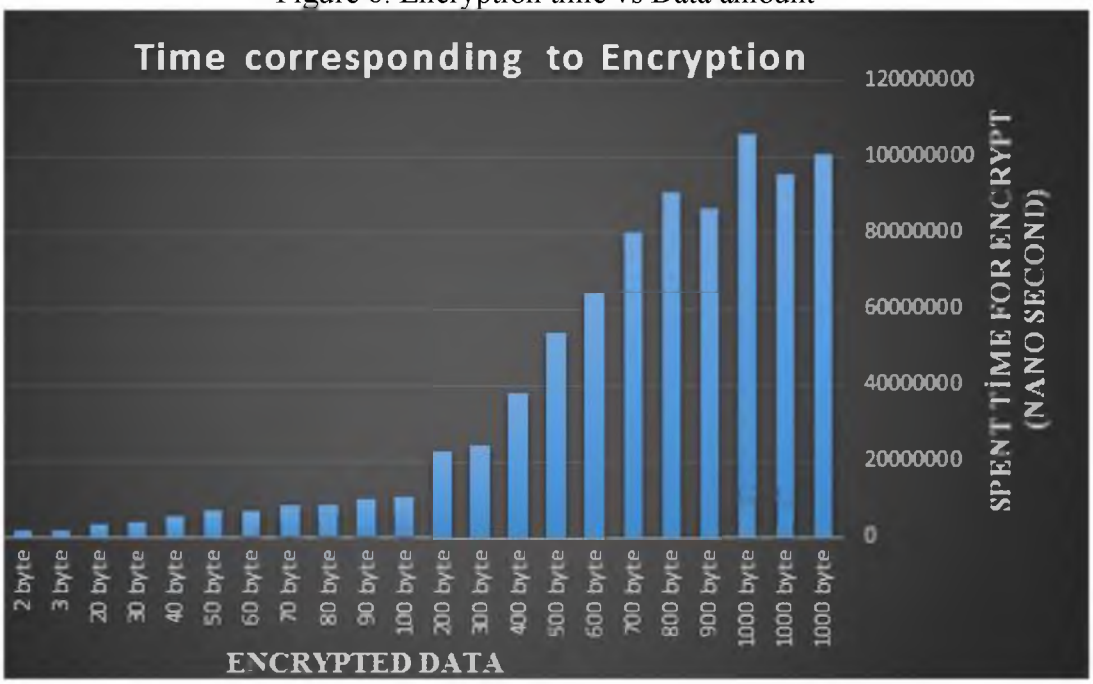
(O) n = 6 + 16 + 1012 + 4 + 1 + 16 + 1 + 16 + 21 + 6 + 18 + 1 + 16 + 1 + 36 + 3 + 16 = 1190 steps.

Hence when implementing AES on the STM32F407VGT6 processor running at 168MHz and assuming that a step code is about 5 nano seconds, the AES encryption process will take roughly 5ns × 1190 = 5950 ns per 128 bits that is 2 bytes.

On the other hand, the same steps are carried out on the smart device running Android, however this is typically faster. In this work, the utilized Android smart phone has an operating frequency of approximately 2300/4 = 600Mhz. Assuming that 1 step is applied directlyeach step will take roughly 1.6 nanoseconds. Hence, approximately 1.6 × 1190 = 1904 nano seconds will be required for AES encryption on the android smartphone per 128 bits that is 2 bytes. These complexity analysis shows thar running AES will take a reasonably small amount of time, and therefore when AES is used, the user is not expected to experience any delay at runtime. Furthermore it is shown in the experimental results below, that the actual time spend is even below these values thanks to optimization possibilities.

Figure 8 shows the time spend for AES encryption against the Data size. While the time might show a small variation depending on the current workload of the device it is seen that even for 1000 Bytes of Data the encryption time is about 0.1 seconds and the encryption time is roughly linear as would be expected. Hence it is seen that a throughput of about 10 kB is achieved in AES encoding. Considering the Data throughput of smart devices[14] this rate is higher than most smart devices can actually achieve and thus will be sufficient for real-time applications.

Figure 8. Encryption time vs Data amount



6. Conclusions

This paper presented the implementation of AES for BLE communication between and embedded system and a smart device. Because BLE facilitates wirless communication, secure and reliable Data transmission is an important aspect for many applications in particular for health and consumer grade devices. It is shown that EAS encryption can be carried out practially in real-time withough imposing a delay or load to the BLE communication for data rates of about 10 kBytes/s. This throughput is acceptable for BLE communication as the data transmission rate is usually limited by the smart devices to around this rate anyway. The presented approach provides secure transmission of Data over BLE ensuring there is no undesired access by third parties.

Reference

1. Announcing the Advanced Encryption Standard (AES), federal Information Processing Standards Publication 197, November 26, 2011 [5]
2. Arif C., "Embedded Cardiac Rhythm Analysis and Wireless Transmission (Wi-CARE)," MS Thesis, School of Computing and Software Engineering, Southern Polytechnic State University, Marietta, Georgia, USA, 2004. [11]
3. Heart Rate Monitoring and Data Transmission via Bluetooth, Prasad Kumari Nisha, Yadav Vinita, 2015 [4]
4. PAST, PRESENT, AND FUTURE METHODS OF CRYPTOGRAPHY AND DATA ENCRYPTION, Nicholas G.McDonald (Department of Electrical and Computer Engineering) [10]
5. Reference manual, STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM – based 32-bit MCUs, RM0090 [7]
6. Sending and Recieving Data via Bluetooth with an Android Device, Brian Wirsing, March 26, 2014 [12]
7. Şifreleme Algoritmalarının Sınıflandırılması ve Algoritmalara Saldırı Teknikleri, Yrd. Doç. Dr. Mehmet Tektaş [6]
8. TinySine Serial Bluetooth 4.0 Smart Ready dual-mode modüle, User manual, HM-12 [9]
9. User manual, Discovery kit for STM32F407/417, UM1472 [8]
10. https://develop.android.com/sdk/index.html [13]
11. https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth [1]
12. https://en.wikipedia.org/wiki/Bluetooth [2]
13. http://www.cypress.com/training/how-make-ios-app-control-robot-using-bluetooth-low-energy-ble [3]
14. https://punchthrough.com/blog/posts/maximizing-ble-throughput-on-ios-and-android [14]